



## CSM-S SSL サービスの設定

この章では、SSL の Content Switching Module with SSL ソフトウェアを設定、モニタ、デバッグするための CLI (コマンドラインインターフェイス) コマンドについて説明します。これらのコンフィギュレーション コマンドは、SSL Services Module (SSLSM) で使用できるコマンドと同じです。



(注)

特に区別されていないかぎり、「*Content Switching Module*」および略語「*CSM*」には、コンテンツスイッチング モジュールと CSM-S の両方が含まれます。「*Content Switching Module with SSL*」および略語「*CSM-S*」は、CSM-S だけを言及する場合に使用されます。

この章では、SSL ドータカードをサポートするために、CSM-S に追加された設定について説明します。

- [SSL ドータカードの初期設定 \(p.7-2\)](#)
- [クライアント側およびサーバ側動作に対する SSL の設定 \(p.7-7\)](#)
- [ポリシーの設定 \(p.7-11\)](#)
- [SSL プロキシ サービスの設定 \(p.7-20\)](#)
- [NAT の設定 \(p.7-25\)](#)
- [TACACS、TACACS+、および RADIUS の設定 \(p.7-26\)](#)
- [SNMP トラップの設定 \(p.7-26\)](#)
- [暗号セルフテストのイネーブル化 \(p.7-28\)](#)
- [クラッシュ情報の収集 \(p.7-31\)](#)
- [VTS デバッグのイネーブル化 \(p.7-33\)](#)



(注)

すべてのバックエンドサーバに別個のサーバファームを作成する必要があります。CSM-S は仮想サーバに関連付けられませんが、CSM-S は各実サーバのアドレス解決を行います。コンフィギュレーション例については、「[バックエンドサーバとしての実サーバの設定](#)」(p.20) を参照してください。

## SSL ドータカードの初期設定

ここでは、SSL ドータカードの初期設定の方法について説明します。



(注) CSM-S 証明書管理ポート コネクタに直接接続して、次の SSL ドータカードの初期設定を行う必要があります (図 1-2[p.1-8] を参照)。この初期設定の完了後は、モジュールに Secure Shell (SSH; セキュア シェル) または Telnet 接続を行い、さらに詳しくモジュールを設定できます。

SSL ドータカードの初期設定の作業は、次のとおりです。

- [SSL ドータカードへの VLAN の設定 \(p.7-2\)](#)
- [Telnet リモート アクセスの設定 \(p.7-3\)](#)
- [FQDN の設定 \(p.7-4\)](#)
- [SSH の設定 \(p.7-4\)](#)

## SSL ドータカードへの VLAN の設定

SSL ドータカードに VLAN (仮想 LAN) を設定する場合は、いずれかの VLAN を管理 VLAN として設定します。管理 VLAN は、SSH、Public Key Infrastructure (PKI; 公開鍵インフラストラクチャ)、Secure File Transfer (SCP) および Trivial File Transfer Protocol (TFTP; 簡易ファイル転送プロトコル) 処理を含む、すべての管理トラフィックに使用されます。管理 VLAN のゲートウェイを通るデフォルトルートが追加されます。



(注) SSL ドータカードには、管理 VLAN として 1 つの VLAN だけを設定してください。



(注) SSL ドータカードに設定されるすべての VLAN は、CSM にも設定される必要があります。すべての VLAN が CSM 仮想サーバと SSL 実サーバで一致する必要があります。




(注) スイッチとモジュールの VLAN ID は同じでなければなりません。詳細については、『*Catalyst 6500 Series Switch Software Configuration Guide*』の「Configuring VLANs」の章を参照してください。



(注) SSL ソフトウェアは、標準範囲の VLAN (2 ~ 1005) だけをサポートします。SSL ドータカードの設定を、標準範囲 VLAN に限定する必要があります。

SSL ドータカードに VLAN を設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy(config)# <b>ssl-proxy vlan</b> vlan</code>	VLAN を設定し、VLAN モードを開始します。
ステップ 2	<code>ssl-proxy(config-vlan)# <b>ipaddr</b> ip_addr netmask</code>	VLAN に IP アドレスを設定します。
ステップ 3	<code>ssl-proxy(config-vlan)# <b>gateway</b> gateway_addr</code>	クライアント側ゲートウェイの IP アドレスを設定します。  (注) VLAN IP アドレスと同じサブネット内のゲートウェイ IP アドレスを設定します。
ステップ 4	<code>ssl-proxy(config-vlan)# <b>route</b> ip_addr netmask gateway ip_addr</code>	(任意) CSM-S からのレイヤ 3 ホップ数が 1 つまたは複数であるサーバに、スタティックルートを設定します。
ステップ 5	<code>ssl-proxy(config-vlan)# <b>admin</b></code>	(任意) VLAN を管理 VLAN <sup>1</sup> として設定します。

1. 管理 VLAN は管理トラフィック (PKI、SSH、SCP、および TFTP) に使用します。管理 VLAN に指定できるのは、1 つの VLAN のみです。

次に、VLAN を設定し、IP アドレス、サブネット マスク、およびグローバル ゲートウェイを指定し、VLAN を管理 VLAN として指定する例を示します。

```
ssl-proxy(config)# ssl-proxy vlan 100
ssl-proxy(config-vlan)# ipaddr 10.1.0.20 255.255.255.0
ssl-proxy(config-vlan)# gateway 10.1.0.1
ssl-proxy(config-vlan)# admin
ssl-proxy(config-vlan)# ^Z
ssl-proxy#
```

## Telnet リモート アクセスの設定

SSL ドータカードを Telnet リモート アクセス用に設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy(config)# <b>enable password</b> password</code>	ローカルなイネーブルパスワードを指定します。
ステップ 2	<code>ssl-proxy(config)# <b>line vty</b> starting-line-number ending-line-number</code>	設定する回線範囲を特定し、ライン コンフィギュレーションモードを開始します。
ステップ 3	<code>ssl-proxy(config-line)# <b>login</b></code>	ログイン時のパスワード検査をイネーブルにします。
ステップ 4	<code>ssl-proxy(config-line)# <b>password</b> password</code>	回線のパスワードを指定します。

次に、SSL ドータカードをリモート アクセス用に設定する例を示します。

```
ssl-proxy(config)# enable password cisco
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)#login
ssl-proxy(config-line)#password cisco
ssl-proxy(config-line)#end
ssl-proxy#
```

## FQDN の設定

SSL ドータカードを使用して認証局からの証明書を登録する場合は、モジュールに Fully Qualified Domain Name (FQDN; 完全修飾ドメイン名) を設定する必要があります。FQDN は、モジュールのホスト名およびドメイン名です。

FQDN を設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy(config)# hostname name</code>	ホスト名を設定します。
ステップ 2	<code>ssl-proxy(config)# ip domain-name name</code>	ドメイン名を設定します。

次に、SSL ドータカードに FQDN を設定する例を示します。

```
ssl-proxy(config)# hostname ssl-proxy2
ssl-proxy2(config)# ip domain-name example.com
ssl-proxy2(config)# end
ssl-proxy2(config)#
```

## SSH の設定

モジュールの初期設定が完了したら、モジュール上で SSH をイネーブルにし、単純なユーザ名とパスワード、または Authentication, Authorization, Accounting (AAA; 認証、許可、アカウントिंग) サーバを使用して、SSH 接続用のユーザ名およびパスワードを設定します。

ここでは、SSH をイネーブルにして設定する手順について説明します。

- [モジュール上での SSH のイネーブル化 \(p.7-4\)](#)
- [SSH 用のユーザ名およびパスワードの設定 \(p.7-5\)](#)
- [SSH 用の AAA の設定 \(p.7-6\)](#)

### モジュール上での SSH のイネーブル化

SSH は、モジュールで生成された最初の鍵ペアを使用します。次の作業では、SSH 専用の鍵ペアを生成します。



(注) 最初に SSH 鍵ペアを指定しないで、汎用の鍵ペアを生成すると（「RSA 鍵のペアの生成」[p.8-5]を参照）、SSH がイネーブルになり、汎用鍵ペアが使用されます。この鍵ペアをあとで削除すると、SSH がディセーブルになります。SSH を再びイネーブルにするには、SSH 鍵ペアを新規に生成します。

SSH 鍵ペアを生成して SSH をイネーブルにする手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy# configure terminal</code>	terminal オプションを選択して、コンフィギュレーションモードを開始します。
ステップ 2	<code>ssl-proxy(config)# ip ssh rsa keypair-name ssh_key_name</code>	SSH に鍵ペアの名前を割り当てます。
ステップ 3	<code>ssl-proxy(config)# crypto key generate rsa general-keys label ssh_key_name</code>	SSH 鍵ペアを生成します。これで、SSH がイネーブルになります。

	コマンド	目的
ステップ 4	<code>ssl-proxy(config)# end</code>	コンフィギュレーションモードを終了します。
ステップ 5	<code>ssl-proxy# show ip ssh</code>	SSH の現在の状態を表示します。

次に、モジュール上で SSH をイネーブルにし、SSH がイネーブルに設定されたことを確認する例を示します。

```
ssl-proxy(config)# ip ssh rsa keypair-name ssh-key
Please create RSA keys to enable SSH.
ssl-proxy(config)# crypto key generate rsa general-keys label ssh-key
The name for the keys will be: ssh-key
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys ...[OK]

ssl-proxy(config)#
*Aug 28 11:07:54.051: %SSH-5-ENABLED: SSH 1.5 has been enabled
ssl-proxy(config)# end

ssl-proxy# show ip ssh
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
ssl-proxy#
```

## SSH 用のユーザ名およびパスワードの設定

SSH 接続用のユーザ名およびパスワードを設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy# configure terminal</code>	terminal オプションを選択して、コンフィギュレーションモードを開始します。
ステップ 2	<code>ssl-proxy(config)# enable password password</code>	ローカルなイネーブルパスワードを（まだ指定されていないならば）指定します。
ステップ 3	<code>ssl-proxy(config)# username username {password   secret} password</code>	ユーザ名およびパスワードを指定します。
ステップ 4	<code>ssl-proxy(config)# line vty line-number ending-line-number</code>	設定する回線範囲を特定し、ライン コンフィギュレーションモードを開始します。
ステップ 5	<code>ssl-proxy(config-line)# login local</code>	ローカルなユーザ名認証をイネーブルにします。

次に、SSL ドータカードに SSH 接続用のユーザ名およびパスワードを設定する例を示します。

```
ssl-proxy# configure terminal
ssl-proxy(config)# enable password cisco
ssl-proxy(config)# username admin password admin-pass
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)# login local
ssl-proxy(config-line)# end
```

ユーザ名およびパスワードを設定したあとに、スイッチを設定する手順については、「[失われたパスワードの回復](#)」(p.3-18) を参照してください。

## SSH 用の AAA の設定

SSH 用の AAA を設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy# configure terminal</code>	terminal オプションを選択して、コンフィギュレーションモードを開始します。
ステップ 2	<code>ssl-proxy(config)# username username secret {0   5} password</code>	指定された検索不能なユーザ名に対して、拡張パスワードセキュリティをイネーブルにします。
ステップ 3	<code>ssl-proxy(config)# enable password password</code>	ローカルなイネーブルパスワードを（まだ指定されていないならば）指定します。
ステップ 4	<code>ssl-proxy(config)# aaa new-model</code>	AAA をイネーブルにします。
ステップ 5	<code>ssl-proxy(config)# aaa authentication login default local</code>	ローカルなユーザ名データベースを認証に使用するように、モジュールを指定します。
ステップ 6	<code>ssl-proxy(config)# line vty line-number ending-line-number</code>	設定する回線範囲を特定し、ライン コンフィギュレーションモードを開始します。
ステップ 7	<code>ssl-proxy(config-line)# transport input ssh</code>	特定の回線で使用する唯一のプロトコルとして SSH を設定します（これにより、非 SSH 接続が禁止されます）。

次に、SSL ドータカードに SSH 接続用の AAA を設定する例を示します。

```
ssl-proxy# configure terminal
ssl-proxy(config)# username admin secret admin-pass
ssl-proxy(config)# enable password enable-pass
ssl-proxy(config)# aaa new-model
ssl-proxy(config)# aaa authentication login default local
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)# transport input ssh
ssl-proxy(config-line)# end
ssl-proxy#
```

AAA を設定したあとに、スイッチを設定する手順については、「[失われたパスワードの回復](#)」(p.3-18) を参照してください。

## クライアント側およびサーバ側動作に対する SSL の設定

ここでは、CSM-S の設定方法について説明します。内容は、次のとおりです。

- [クライアント側の設定 \(p.7-7\)](#)
- [サーバ側の設定 \(p.7-8\)](#)

サーバ ファームを設定するときに、実サーバが SSL ドータカードである場合は、実サーバの定義に **local** キーワードを使用する必要があります。

次に、SSL をサポートするように、CSM を設定する例を示します。

```
Cat6k-2 (config-module-csm) # serverfarm SSLfarm
Cat6k-2 (config-slb-sfarm) # real 10.1.0.21 local
Cat6k-2 (config-slb-real) # inservice

Cat6k-2 (config-module-csm) # vserver VS1
Cat6k-2 (config-slb-vserver) # virtual 10.1.0.21 tcp https
Cat6k-2 (config-slb-vserver) # serverfarm SSLfarm
Cat6k-2 (config-slb-vserver) # inservice
```

実サーバへの **local** キーワードは、CSM に対する唯一の設定変更です。CSM と SSL ドータカード間で適切に通信が行われるように、さらに詳しい設定を CSM-S に行う必要があります。

## クライアント側の設定

次に、SSL ドータカードに SSL プロキシ サービスを設定する例を示します。

```
ssl-proxy(config) # ssl-proxy service S1
ssl-proxy(config-ssl-proxy) # virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy) # server ipaddr 10.2.0.100 protocol TCP port 80
ssl-proxy(config-ssl-proxy) # inservice
```

次に、CSM 仮想サーバの設定例を示します。

```
Cat6k-2 (config-module-csm) # serverfarm SSLfarm
Cat6k-2 (config-slb-sfarm) # real 10.1.0.21 local
Cat6k-2 (config-slb-real) # inservice

Cat6k-2 (config-module-csm) # vserver VS1
Cat6k-2 (config-slb-vserver) # virtual 10.1.0.21 tcp https
Cat6k-2 (config-slb-vserver) # serverfarm SSLfarm
Cat6k-2 (config-slb-vserver) # inservice
```

混合モードの SSL ドータカードと SSL サービス モジュール間で SSL ロードバランシングを実行できます。

CSM は SSL-ID 固定 (sticky) 機能を使用して、同じ SSL サービス モジュールに SSL 接続を固定します。CSM は、SSL-ID を調べるためにクライアント側の TCP 接続を終了する必要があります。ロードバランシングの決定が行われたら、CSM は SSL ドータカードまたは SSL サービス モジュールのどちらかに対して TCP 接続を開始する必要があります。

トラフィック フローは、CSM が仮想サーバで受信されたすべてのトラフィックを SSL ドータカードに送り、SSL ドータカード自体で TCP を終了させる形を取ります。SSL スティッキー機能をイネーブルにする場合、CSM と SSL ドータカード間の接続は、完全な TCP 接続になります。

次に、混在モードの SSL ロードバランシングを設定する例を示します。

```
Cat6k-2(config-module-csm)# sticky 10 ssl timeout 60
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-slb-sfarm)# real 10.2.0.21
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
Cat6k-2(config-slb-vserver)# sticky 60 group 10
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# persistent rebalance
Cat6k-2(config-slb-vserver)# inservice
```

さらに、CSM がクライアント側の TCP 接続を終了する必要がある場合に、SSL ドータカードにトラフィックを送るように、内部で生成される設定を行う必要があります。サーバファーム *SSLfarm* のローカルな各実サーバの同じ IP アドレスまたはポートを指定して、仮想サーバを作成する必要があります。この仮想サーバ向けのすべてのトラフィックを SSL ドータカードに送るように、この仮想サーバが内部的に設定されます。

ローカル実サーバの IP アドレスと SSL ドータカード仮想サーバのアドレスを一致させなければならないため、内部生成の形でコンフィギュレーションを作成する必要があります。CSM がこのローカル実サーバへの接続を開始すると、SYN フレームが CSM で送受信されます。CSM が SYN を受信し、宛先 IP アドレスまたはポートが仮想サーバ VS1 と同一である場合、より詳細な仮想サーバが追加されていないかぎり、VS1 に一致します。

## サーバ側の設定

SSL ドータカードが CSM をバックエンドサーバとして使用する場合、標準の仮想サーバの設定がレイヤ 4 およびレイヤ 7 のロードバランシング用に使用されます。

この仮想サーバが SSL ドータカードからのトラフィックだけを受信するように制限するには、次のように VLAN ローカル仮想サーバサブモード コマンドを使用します。

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-sfarm)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

次の例のように、実サーバをバックエンドとして設定できます。

```
Cat6k-2(config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-sfarm)# predictor forward

Cat6k-2(config-module-csm)# vserver VS3
Cat6k-2(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-vserver)# inservice
```



## バックエンド サーバとしての CSM の設定

仮想サーバおよびサーバ ファームの設定により、SSL ドータカードがバックエンドサーバとして実サーバを使用することができます。CSM をバックエンドサーバとして使用するには、「[クライアント側の設定](#)」(p.7-7) で説明した設定を使用し、さらに SSL ドータカードを設定します。

次に、レイヤ 7 のロードバランシングに対応する CSM 仮想サーバの設定例を示します。

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# serverfarm SLBjpgfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.21

Cat6k-2(config-module-csm)# map JPG url
Cat6k-2(config-slb-map-cookie)# match protocol http url *jpg*

Cat6k-2(config-module-csm)# policy SLBjpg
Cat6k-2(config-slb-policy)# url-map JPG
Cat6k-2(config-slb-policy)#serverfarm SLBjpgfarm

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# slb-policy SLBjpg
Cat6k-2(config-slb-vserver)# inservice
```

次に、レイヤ 4 のロードバランシングに対応する CSM 仮想サーバの設定例を示します。

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

## バックエンド サーバとしての実サーバの設定

サーバ側の設定によるトラフィック フロー（バックエンドとして実サーバを設定する場合）は、クライアント側の設定と類似しています。「[クライアント側の設定](#)」(p.7-7)に記載されている設定を使用してから、SSL ドータカードがバックエンドサーバとして実サーバを使用するように設定します。

SSL ドータカード プロキシ サービスの設定には、新たな設定が必要ありません。次に、設定を内部で開始し、ユーザからはわからないようにする例を示します。

```
ssl-proxy(config)# ssl-proxy service S1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 10.2.0.20 protocol TCP port 80
ssl-proxy(config-ssl-proxy)# inservice
```

次に、CSM 仮想サーバの設定例を示します。

```
Cat6k-2 (config-module-csm)# serverfarm SSLreals

Cat6k-2 (config-slb-sfarm)# real 10.2.0.20
Cat6k-2 (config-slb-sfarm)# inservice

Cat6k-2 (config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2 (config-slb-sfarm)# predictor forward

Cat6k-2 (config-module-csm)# vserver VS3
Cat6k-2 (config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2 (config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2 (config-slb-vserver)# inservice
```

## ポリシーの設定

ここでは、SSL ポリシーおよび TCP ポリシーの設定方法について説明します。

- [SSL ポリシーの設定 \(p.7-11\)](#)
- [TCP ポリシーの設定 \(p.7-13\)](#)
- [HTTP ヘッダー挿入 \(p.7-14\)](#)
- [URL リライトの設定 \(p.7-17\)](#)

## SSL ポリシーの設定



(注) SSL ドータカードの SSL コマンドは、グローバルにも、また特定のプロキシ サーバにも適用されます。プロキシ サービスにポリシーを適用する手順については、「[SSL サーバ プロキシ サービス \(p.7-20\)](#)」を参照してください。

SSL ポリシー テンプレートを使用すると、SSL スタックに関連付けられたパラメータを定義することができます。

設定できるパラメータの 1 つに、SSL 終了プロトコルの動作があります。SSL 終了プロトコルでは、各 SSL ピア (クライアント / サーバ) が、終了通知アラートの送信および終了通知アラートの受信を行ってから、接続を適切に終了する必要があることを指定します。SSL 接続が適切に終了しない場合は、セッションが削除されるので、以後の SSL 接続で同一の SSL セッション ID を使用することができません。

ただし、厳密には SSL 終了プロトコルより先に処理される SSL 動作は多数あります (たとえば、SSL ピアは終了通知アラートを送信しますが、リモート SSL ピアからの終了通知アラートを待機せずに、接続を終了します)。


SSL ピアが終了接続シーケンスを開始すると、SSL ドータカードは終了通知アラートメッセージを待機します。SSL ピアが終了通知アラートを送信しない場合、SSL ドータカードはセッション キャッシュからセッションを削除するので、以後の SSL 接続で同一のセッション ID を使用することができません。

SSL ドータカードが終了接続シーケンスを開始する場合、次の終了プロトコル オプションを設定できます。

- **strict** — SSL ドータカードは SSL ピアに終了通知アラートメッセージを送信し、SSL ドータカードは SSL ピアからの終了通知アラートメッセージを待機します。SSL ドータカードが終了通知アラートを受信しなければ、そのセッションの SSL の再開が許可されません。
- **none** — SSL ドータカードは SSL ピアに終了通知アラートメッセージを送信せず、SSL ドータカードは SSL ピアからの終了通知アラートメッセージを待機しません。SSL ドータカードが SSL ピアから終了通知アラートを受信すると、SSL ドータカードはセッション情報を維持するので、以後の SSL 接続で SSL の再開を使用できます。ただし、SSL ドータカードが SSL ピアから終了通知アラートを受信しなければ、そのセッションの SSL の再開が許可されません。
- **disabled** (デフォルト) — SSL ドータカードは SSL ピアに終了通知アラートを送信しますが、SSL ピアは終了通知アラートを待機せずに、セッションを削除します。SSL ピアが終了通知アラートを送信するかどうかに関係なく、セッション情報は維持され、以後の SSL 接続でセッションの再開が許可されます。

SSL ポリシーを特定のプロキシ サーバに関連付けない場合は、プロキシ サーバはデフォルトで、サポート対象の暗号スイートとプロトコルバージョンをすべてイネーブルにします。

SSL ポリシー テンプレートを定義して、特定のプロキシ サーバに SSL ポリシーを関連付ける手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy (config)# ssl-proxy policy ssl ssl_policy_name</code>	SSL ポリシー テンプレートを定義します。
ステップ 2	<code>ssl-proxy (config-ssl-policy)# cipher {rsa-with-rc4-128-md5   rsa-with-rc4-128-sha   rsa-with-des-cbc-sha   rsa-with-3des-ede-cbc-sha   others...}</code>	プロキシ サーバが受け入れ可能な暗号スイートの名前リストを設定します。暗号スイート名の表記法は、既存の SSL スタックの表記法と同じです。
ステップ 3	<code>ssl-proxy (config-ssl-policy)# protocol {ssl3   tls1   all}</code>	プロキシ サーバでサポートされる各種プロトコルバージョンを定義します。
ステップ 4	<code>ssl-proxy (config-ssl-policy)# timeout handshake time</code>	モジュールがハンドシェイク フェーズで接続を維持できる時間を設定します。有効範囲は 0 ~ 65535 秒です。
ステップ 5	<code>ssl-proxy (config-ssl-policy)# close-protocol {strict   none}</code>	SSL 終了プロトコルの動作を設定します。終了プロトコルの動作は、デフォルトでディセーブルです。
ステップ 6	<code>ssl-proxy (config-ssl-policy)# session-cache</code>	セッションキャッシュ機能をイネーブルにします。セッション キャッシュはデフォルトでイネーブルです。
ステップ 7	<code>ssl-proxy (config-ssl-policy)# timeout session timeout [absolute<sup>1</sup>]</code>	<p>エントリがセッション キャッシュ内に維持される時間を設定します。有効範囲は 1 ~ 72000 秒です。</p> <p> (注) セッションキャッシュ サイズを設定するには、<b>absolute</b> キーワードが必要です。<b>absolute</b> キーワードは、セッション エントリが指定された <i>timeout</i> の間セッション キャッシュ内に維持されるよう指定します。<b>absolute</b> キーワードが指定されているときは、セッション キャッシュ内に利用できるフリー エントリがない場合、新たな着信接続は拒否されます。</p>
ステップ 8	<code>ssl-proxy (config-ssl-policy)# session-cache size size</code>	<p>(任意)セッション キャッシュのサイズを指定します<sup>1</sup>。有効範囲は 1 ~ 262143 エントリです。</p> <p> (注) <b>timeout session</b> コマンドで <b>absolute</b> キーワードを入力して、セッション キャッシュのサイズを指定します。このコマンドを入力しない場合、または <i>size</i> が指定されていない場合は、セッション キャッシュのサイズは最大サイズになります (262,144)。</p>

1. **absolute** キーワードが設定されると、設定されたセッションがタイムアウトになってから、セッション エントリが再利用されます。**absolute** が設定されると、必要なセッション エントリ数は (`new_connection_rate * absolute_timeout`) と同等になります。タイムアウト設定および新しい接続レートに応じて、セッション エントリ数が大きくなる場合があります。セッションキャッシュ サイズを設定することにより、セッション エントリ数を制限できます。

## TCP ポリシーの設定



(注) SSL ドータカードの TCP コマンドは、グローバルにも、また特定のプロキシサーバにも適用されます。

TCP ポリシー テンプレートをを使用すると、TCP スタックに関連付けられたパラメータを定義することができます。

TCP ポリシー テンプレートを定義して、特定のプロキシサーバに TCP ポリシーを関連付ける手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy (config)# ssl-proxy policy tcp tcp_policy_name</code>	TCP ポリシー テンプレートを定義します。特に指定がない場合は、デフォルト設定がすべて適用されます。
ステップ 2	<code>ssl-proxy (config-tcp-policy)# timeout syn time</code>	接続確立タイムアウトを設定します。デフォルト値は 75 秒です。有効範囲は 5 ~ 75 秒です。
ステップ 3	<code>ssl-proxy (config-tcp-policy)# mss max_segment_size</code>	生成された SYN パケット内で接続が識別する Maximum Segment Size (MSS; 最大セグメント サイズ) をバイト単位で設定します。   (注) このコマンドでは、プロキシサーバのクライアント側およびサーバ側に異なる MSS を設定できます。デフォルト値は 1460 バイトです。有効範囲は 256 ~ 2460 バイト <sup>1</sup> です。
ステップ 4	<code>ssl-proxy (config-tcp-policy)# timeout reassembly time</code>	リアセンブリ キューが削除されるまでの時間を秒単位で設定します。トランザクションが指定時間内で完了されない場合は、リアセンブリ キューが削除され、接続がドロップされます。デフォルト値は 60 秒です。有効範囲は 0 ~ 960 秒 (0 = デイセーブル) です。
ステップ 5	<code>ssl-proxy (config-tcp-policy)# timeout inactivity time</code>	確立された接続が非アクティブである時間を秒単位で設定します。デフォルト値は 600 秒です。有効範囲は 0 ~ 960 秒 (0 = デイセーブル) です。
ステップ 6	<code>ssl-proxy (config-tcp-policy)# timeout fin-wait time</code>	FIN 待機タイムアウトを秒単位で設定します。デフォルト値は 600 秒です。有効範囲は 75 ~ 600 秒です。
ステップ 7	<code>ssl-proxy (config-tcp-policy)# buffer-share rx buffer_limit</code>	接続ごとに共有する最大受信バッファをバイト単位で設定します。デフォルト値は 32768 バイトです。有効範囲は 8192 ~ 262144 バイトです。
ステップ 8	<code>ssl-proxy (config-tcp-policy)# buffer-share tx buffer_limit</code>	接続ごとに共有する最大送信バッファをバイト単位で設定します。デフォルト値は 32768 バイトです。有効範囲は 8192 ~ 262144 バイトです。

1. フラグメンテーションが発生する場合は、フラグメンテーションが解消されるまで MSS 値を削減してください。

## HTTP ヘッダー挿入

SSL オフロード環境では、SSL オフローダはセキュア クライアント HTTP (HTTPS) 接続を終了し、SSL トラフィックをクリア テキストに復号化し、HTTP 接続経由でクリア テキストを Web サーバに転送します。HTTPS 接続は、クライアント接続が安全な接続として送信されたことを認識しないので、バックエンドサーバで安全でない HTTP 接続になります。

HTTP ヘッダー挿入を設定する理由は、次のとおりです。

- HTTP ヘッダー挿入により、SSL ドータカードはクライアント接続中に、HTTP ヘッダーに情報を組み込むことができます。バックエンドサーバがこのヘッダーを認識すると、すべての URL を HTTPS として戻します。
- クライアントから受信した HTTP ヘッダーにクライアント証明書情報を挿入するように、SSL オフローダを設定することにより、バックエンドアプリケーションで接続ごとの情報を記録することができます。
- SSL ドータカードをサイト間の設定で使用して、安全なチャネルでトラフィックを送信する場合、接続のサーバ側でクライアント IP アドレスとポート情報 (NAT 中に削除される) を認識する必要がある場合があります。

HTTP ヘッダー挿入は、GET、HEAD、PUT、TRACE、POST、および DELETE の方式で実行されません。HTTP ヘッダー挿入は、CONNECT 方式では実行されません。

カスタム ヘッダー、クライアント IP ヘッダーおよびポート ヘッダーは、HTTP 要求パケットごとに挿入されます。すべてのセッションヘッダーおよびデコードされたクライアント証明書フィールドは、最初の HTTP 要求パケットで挿入されます。同じセッション ID を使用する後続の HTTP 要求で挿入されるのは、セッション ID だけです。サーバはセッションおよびクライアント証明書ヘッダーを取得するために、セッション ID に基づいてセッションまたはクライアント証明書ヘッダーをキャッシュし、後続の要求でセッション ID を使用するように見込まれます。

最大 100 の HTTP ヘッダー挿入ポリシーを設定できます。各ポリシーには、最大 32 のプレフィクスまたはヘッダーが含まれます。プレフィクスとカスタム ヘッダーには、最大 240 の文字を含めることができます。

ここでは、HTTP ヘッダーに挿入することができる情報について説明します。

- [プレフィクス \(p.7-14\)](#)
- [クライアント証明書ヘッダー \(p.7-15\)](#)
- [クライアント IP およびポートアドレスヘッダー \(p.7-15\)](#)
- [カスタムヘッダー \(p.7-16\)](#)
- [SSL セッションヘッダー \(p.7-16\)](#)

## プレフィクス

`prefix prefix_string` を指定すると、SSL ドータカードは、すべての挿入された HTTP ヘッダーに指定されたプレフィクスを追加します。プレフィクスを追加することにより、サーバは、他の装置からではなく SSL ドータカードからの接続であることを識別できるようになります。プレフィクスは、クライアントからの標準の HTTP ヘッダーには追加されません。`prefix_string` には、最大 240 の文字を加えることができます。

## クライアント証明書ヘッダー

クライアント証明書ヘッダー挿入により、バックエンドサーバは、SSL ドータカードが認証および承認したクライアント証明書の属性を参照できます。クライアント証明書ヘッダーは、セッションにつき1回だけ送信されます。サーバは、セッション ID（ヘッダーと同様に挿入される）を使用して、これらの値をキャッシュするよう見込まれます。後続の要求では、サーバはセッション ID を使用して、サーバ自体にキャッシュされたクライアント証明書ヘッダーを検索します。



(注) クライアントが証明書を送信しない場合、SSL ハンドシェイクが失敗します。データ フェーズまたはヘッダー挿入も行われません。

**client-cert** を指定すると、SSL ドータカードが次のヘッダーをバックエンドサーバに送ります。

挿入するフィールド	説明
ClientCert-Valid	証明書の有効性
ClientCert-Error	エラー条件
ClientCert-Fingerprint	ハッシュ出力
ClientCert-Subject-CN	X.509 所有者の一般名
ClientCert-Issuer-CN	X.509 証明書発行人の一般名
ClientCert-Certificate-Version	X.509 証明書のバージョン
ClientCert-Serial-Number	証明書のシリアル番号
ClientCert-Data-Signature-Algorithm	X.509 のハッシュおよび暗号化方式
ClientCert-Subject	X.509 所有者の識別名
ClientCert-Issuer	X.509 証明書発行人の識別名
ClientCert-Not-Before	証明書の有効期間開始日
ClientCert-Not-After	証明書の有効期間終了日
ClientCert-Public-Key-Algorithm	公開鍵に使用されるアルゴリズム
ClientCert-RSA-Public-Key-Size	RSA 公開鍵のサイズ
ClientCert-RSA-Modulus-Size	RSA 秘密鍵のサイズ
ClientCert-RSA-Modulus	RSA 係数
ClientCert-RSA-Exponent	RSA 公開指数
ClientCert-X509v3-Authority-Key-Identifier	X.509 認証鍵の識別子
ClientCert-X509v3-Basic-Constraints	X.509 基本制約
ClientCert-Signature-Algorithm	証明書の署名アルゴリズム
ClientCert-Signature	証明書の署名

## クライアント IP およびポート アドレス ヘッダー

Network Address Translation (NAT; ネットワーク アドレス変換) は、クライアントの IP アドレスと宛先 TCP ポート番号情報を変更します。**client-ip-port** を指定すると、SSL ドータカードは HTTP ヘッダーにクライアント IP アドレスと TCP 宛先ポート情報を挿入するので、サーバはクライアント IP アドレスと宛先ポート番号を参照できます。

## カスタム ヘッダー

`custom custom_string` を指定すると、SSL ドータカードは、ユーザ定義のヘッダーをそのまま HTTP ヘッダーに挿入します。HTTP ヘッダー ポリシーごとに、最大 16 のカスタム ヘッダーを設定できます。`custom_string` には、最大 240 の文字を含めることができます。



(注) `custom_string` の構文は、`name:value` の形式です。次のようなスペースが含まれる場合、`custom_string` を引用符で囲む必要があります。

“SOFTWARE VERSION 2.1(1)”

## SSL セッション ヘッダー

セッション ID を含むセッションヘッダーは、セッション ID に基づいてクライアント証明書をキャッシュするのに使用されます。またセッションヘッダーは、サーバが特定の暗号スイートに基づく接続を追跡する場合にも、セッション ID に基づいてキャッシュされます。SSL ドータカードは、すべての SSL ハンドシェイクの間は、HTTP 要求にすべてのセッションヘッダーを挿入しますが、セッションが再開すると、セッション ID だけを挿入します。

SSL ドータカードをクライアントとして設定する場合、SSL ドータカードは、モジュールとバックエンド SSL サーバ間の接続のセッション ID を挿入します。

`session` を指定すると、SSL ドータカードは、次のセッションヘッダーの形式で SSL 接続に特有の情報をバックエンドサーバに送ります。

挿入するフィールド	説明
Session-Id	SSL セッション ID
Session-Cipher-Name	対称暗号スイート
Session-Cipher-Key-Size	対称暗号鍵のサイズ
Session-Cipher-Use-Size	対称暗号の使用

## HTTP ヘッダー挿入の設定

HTTP ヘッダー挿入を設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy (config)# ssl-proxy policy http-header policy_name</code>	HTTP ヘッダー挿入を設定します。
ステップ 2	<code>ssl-proxy (config-http-header-policy)# {prefix prefix_string   client-cert   client-ip-port   custom custom_string}   session]</code>	ヘッダーのプレフィクスまたはタイプを指定します。
ステップ 3	<code>ssl-proxy (config-http-header-policy)# exit</code>	コンフィギュレーションモードに戻ります。
ステップ 4	<code>ssl-proxy (config)# ssl-proxy service service_name</code>	SSL プロキシ サービスの名前を定義します。
		<p>(注) <code>service_name</code> 値は大文字と小文字の区別があります。</p>
ステップ 5	<code>ssl-proxy (config-ssl-proxy)# policy http-header http_header_policy_name</code>	要求に対して、プロキシサーバに HTTP ヘッダー ポリシーを適用します。



次に、プレフィクスとセッションヘッダーを挿入するように、SSL ドータカードを設定する例を示します。

```
ssl-proxy (config)# ssl-proxy policy http-header ssl-offload
ssl-proxy (config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy (config-http-header-policy)# session
ssl-proxy (config-http-header-policy)# custom "SOFTWARE VERSION:2.1(1) "
ssl-proxy (config-http-header-policy)# custom "module:SSL MODULE - CATALYST 6500"
ssl-proxy (config-http-header-policy)# custom
type-of-proxy:server_proxy_1024_bit_key_size
ssl-proxy (config-http-header-policy)# exit
ssl-proxy (config)# ssl-proxy service ssl-offload
ssl-proxy (config-ssl-proxy)# policy http-header ssl-offload
```

標準の HTTP ヘッダーに加えて、次のヘッダー情報が挿入されます。

```
SSL-OFFLOAD-SOFTWARE VERSION:2.1(1)
SSL-OFFLOAD-module:SSL MODULE - CATALYST 6500
SSL-OFFLOAD-type-of-proxy:server_proxy_1024_bit_key_size
SSL-OFFLOAD-Session-Id:33:FF:2C:2D:25:15:3C:50:56:AB:FA:5A:81:0A:EC:E9:00:00:0A:03:00:
60:
 2F:30:9C:2F:CD:56:2B:91:F2:FF
SSL-OFFLOAD-Session-Cipher-Name:RC4-SHA
SSL-OFFLOAD-Session-Cipher-Key-Size:128
SSL-OFFLOAD-Session-Cipher-Use-Size:128
```

## URL リライトの設定

一般的な SSL オフロード環境では、SSL オフローダはセキュアクライアント HTTP (HTTPS) 接続を終了し、SSL トラフィックをクリアテキストに復号化し、HTTP 接続経由でクリアテキストを Web サーバに転送します。HTTPS 接続は、クライアント接続が安全な接続として送信されたことを認識しないため、バックエンドサーバで安全でない HTTP 接続になります。

クライアントに戻されたデータに HTTP リダイレクトリンクが含まれ、クライアントがこのリンクに従う場合、クライアントは安全なドメインを離れるので、安全な接続でなくなります。クリアテキスト接続では、リダイレクトされたリンクをサーバからは入手できない可能性があります。

1 つまたは複数の URL リライトルールを設定すれば、バックエンドサーバからの安全でない HTTP リダイレクトによるこのような問題を回避できます。各リライトルールは、SSL プロキシリストのサービスに関連付けられます。URL リライトルールは、http:// から https:// にドメインをリライトすることで安全でない HTTP URL をリダイレクトしてしまう Web サイトの問題を解決します。URL リライトを設定すると、Web サーバへのすべてのクライアント接続が SSL 接続となり、HTTPS コンテンツのクライアントへの安全な返信が保証されます。



(注) URL リライトは、リダイレクトリンクのリライトをサポートします。システムは、サーバからの応答の HTTP ヘッダーフィールドの [Location:] だけをスキャンし、それによってルールをリライトします。URL リライトは、組み込みリンクをサポートしません。


URL リライトは、プロトコルおよび非デフォルトポート (デフォルトポートはクリアテキストではポート 80、SSL ではポート 443) をリライトします。

最大 100 の URL リライトポリシーを設定できます。各ポリシーには、SSL ポリシーサービスごとに最大 32 のリライトルール、ルールごとに最大 200 の文字が含まれます。

URL リライトの注意事項は、次のとおりです。

- URL の完全一致は、ワイルドカードルールより優先されます。サフィクス ワイルドカードルールは、プレフィクス ワイルドカードルールより優先されます。  
たとえば、**www.cisco.com** が最優先され、次に **www.cisco.\***、および **\*.cisco.com** の順になります。
- 1 回につき 1 つのサフィクスまたはプレフィクス ワイルドカードルールだけを入力します。たとえば、同一ポリシー内に **www.cisco.\*** と **www.cisco.c\***、または **\*w.cisco.com** と **\*.cisco.com** を入力しないでください。
- 同一ポリシー内に、2 つの URL 完全一致ルールを入力しないでください。たとえば、同一ポリシー内に **www.cisco.com clearport 80 sslport 443** と **www.cisco.com clearport 81 sslport 444** を入力しないでください。この場合、2 番目のルールが最初のルールを上書きします。
- URL リライトは、オフロードおよびバックエンド サーバの両方 (HTTP から HTTPS、および HTTPS から HTTP) で実行されます。ここでは、ポートのリライトも含まれます。

URL リライトを設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy(config)# <b>ssl-proxy policy</b> url-rewrite policy_name</code>	URL リライト ポリシーを設定します。
ステップ 2	<code>ssl-proxy(config-url-rewrite-policy) # url url [clearport port_number]<sup>1, 2</sup> [sslport port_number]</code>	URL リライト ルールを指定します。1 つの SSL プロキシ サービスに対して最大 32 のリライト ルール、1 つのルールに対して最大 240 の文字を設定できます。   (注) リライトルールごとに、サフィクスまたはプレフィクス ワイルドカード文字 (*) を 1 回ずつ 1 字のみ入力してください。
ステップ 3	<code>ssl-proxy(config-url-rewrite-policy) # exit</code>	コンフィギュレーション モードに戻ります。
ステップ 4	<code>ssl-proxy(config)# <b>ssl-proxy service</b> service_name</code>	SSL プロキシ サービスの名前を定義します。   (注) <i>service_name</i> 値は大文字と小文字の区別があります。
ステップ 5	<code>ssl-proxy(config-ssl-proxy)# <b>policy</b> url-rewrite policy_name</code>	URL リライト ポリシーを適用します。

1. **clearport port\_number** は、リライトされる URL のポート部分を指定します。デフォルトのクリアテキストポート 80 でない場合は、**clearport port\_number** を指定します。
2. **sslport port\_number** は、リライトされる URL のポート部分を指定します。デフォルトの SSL ポート 443 でない場合は、**sslport port\_number** を指定します。

次に、URL リライト ポリシーを設定し、プロキシ サービスにこのポリシーを適用する例を示します。

```
ssl-proxy(config)# ssl-proxy policy url-rewrite cisco_url
ssl-proxy(config-ssl-proxy)# url www.cisco.*
ssl-proxy(config-ssl-proxy)# url www.cisco.com clearport 81 sslport 444
ssl-proxy(config-ssl-proxy)# url wwwin.cisco.com clearport 81 sslport 440
ssl-proxy(config-ssl-proxy)# url 10.1.1.10 clearport 81 sslport 444
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)# ssl-proxy service cisco_service
ssl-proxy(config-ssl-proxy)# policy url-rewrite cisco_url
```

URL リライトの例については、表 7-1 を参照してください。

表 7-1 サーバ プロキシのルールと結果

URL リライト ルール	一致する URL	URL リライト
url www.cisco.com	http://www.cisco.com/	https://www.cisco.com/
url www.cisco.com clearport 81	http://www.cisco.com:81/	https://www.cisco.com/
url www.cisco.com sslport 444	http://www.cisco.com/	https://www.cisco.com:444/
url www.cisco.com clearport 81 sslport 444	http://www.cisco.com:81/	https://www.cisco.com:444/

## SSL プロキシ サービスの設定

SSL プロキシ サービスを定義するには、`ssl-proxy service ssl_proxy_name` コマンドを使用します。プロキシ サービスに関連付けられた仮想 IP アドレス、ポート、および対応するターゲット IP アドレスとポートを設定できます。

プロキシのクライアント側 (**virtual**) およびサーバ側 (**server**) の両方に、TCP ポリシーおよび SSL ポリシーを定義できます。

ここでは、プロキシ サービスを設定する手順について説明します。

- [SSL サーバプロキシ サービス \(p.7-20\)](#)
- [SSLv2 転送 \(p.7-22\)](#)
- [SSL クライアントプロキシ サービス \(p.7-23\)](#)

### SSL サーバ プロキシ サービス

SSL サーバプロキシ サービスを設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy(config)# ssl-proxy service service_name</code>	SSL プロキシ サービスの名前を定義します。  (注) <code>service_name</code> 値は大文字と小文字の区別があります。
ステップ 2	<code>ssl-proxy(config-ssl-proxy)# virtual ipaddr ip_addr [mask_addr]<sup>1,2</sup> protocol tcp port port {secondary}</code>	CSM-S がプロキシとして機能する仮想サーバの IP アドレス、トランスポートプロトコル (TCP) およびポート番号を定義します。  (注) <code>secondary</code> キーワードは、必ず必要となります。
ステップ 3	<code>ssl-proxy(config-ssl-proxy)# server ipaddr ip_addr protocol tcp port port</code>	プロキシのターゲットサーバの IP アドレス、ポート番号、およびトランスポートプロトコルを定義します。  (注) ターゲットサーバの IP アドレスには、SLB デバイスの仮想 IP アドレス、または Web サーバの実 IP アドレスを指定できます。
ステップ 4	<code>ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp_policy_name<sup>3</sup></code>	(任意) プロキシサーバのクライアント側に TCP ポリシーを適用します。TCP ポリシーのパラメータについては、「 <a href="#">TCP ポリシーの設定</a> 」(p.7-13) を参照してください。
ステップ 5	<code>ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl_policy_name<sup>3</sup></code>	(任意) プロキシサーバのクライアント側に SSL ポリシーを適用します。SSL ポリシーのパラメータについては、「 <a href="#">SSL ポリシーの設定</a> 」(p.7-11) を参照してください。
ステップ 6	<code>ssl-proxy(config-ssl-proxy)# server policy tcp tcp_policy_name</code>	(任意) プロキシサーバのサーバ側に TCP ポリシーを適用します。「 <a href="#">TCP ポリシーの設定</a> 」(p.7-13) を参照してください。

	コマンド	目的
ステップ 7	<code>ssl-proxy(config-ssl-proxy)# policy http-header http_header_policy_name</code>	(任意) プロキシ サーバに HTTP ヘッダー ポリシーを適用します。「HTTP ヘッダー挿入」(p.7-14) を参照してください。
ステップ 8	<code>ssl-proxy(config-ssl-proxy)# policy url-rewrite url_rewrite_policy_name</code>	(任意) URL リライト ポリシーを適用します。「URL リライトの設定」(p.7-17) を参照してください。
ステップ 9	<code>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</code>	(任意) プロキシ サービスに信頼される認証局プールを関連付けます。認証局プールについては、「クライアントの証明書の認証」(p.8-47) を参照してください。
ステップ 10	<code>ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only<sup>4</sup>   all<sup>5</sup>}</code>	(任意) サーバの証明書認証をイネーブルにして、検証の形式を指定します。サーバ証明書認証については、「サーバの証明書の認証」(p.8-50) を参照してください。
ステップ 11	<code>ssl-proxy(config-ssl-proxy)# nat {server   client natpool_name}</code>	(任意) CSM-S によってオープンされたサーバ側の接続に、サーバ NAT <sup>6</sup> またはクライアント NAT を使用するように指定します。「NAT の設定」(p.7-25) および「NAT の設定」(p.7-25) を参照してください。
ステップ 12	<code>ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint trustpoint_label</code>	<p>トラストポイントの設定をプロキシ サーバに適用します<sup>7</sup>。</p> <p> (注) トラストポイントは認証局サーバ、鍵パラメータと鍵生成方法、およびプロキシ サーバの証明書登録方法を定義します。トラストポイントの設定手順については、「トラストポイントの宣言」(p.8-7) を参照してください。</p>
ステップ 13	<code>ssl-proxy(config-ssl-proxy)# inservice</code>	プロキシ サーバを管理上のアップに設定します。

1. マスク アドレスを設定して、ワイルドカードのプロキシ サービスを指定します。ワイルドカードのプロキシ サービスを設定するには、**secondary** キーワードを入力する必要があります。
2. **secondary** キーワードを入力すると、SSL ドータカードは仮想 IP アドレスの ARP 要求に応答しません。
3. パラメータを指定しないでポリシーを作成すると、デフォルト値を使用してポリシーが作成されます。
4. **signature-only** を確認する場合、信頼される認証局プールの信頼される認証局トラストポイントの 1 つに対応するレベルで、認証が停止します。
5. すべてを検証する場合は、証明書チェーンの最上位の発行元が信頼できる認証局トラストポイントとして設定されていなければなりません。SSL ドータカードは、ピア証明書チェーンに含まれるすべての証明書を認証し、最上位の認証局でのみ停止します。最上位の認証局に対応する認証局トラストポイントがなければなりません。また、このトラストポイントを確認する必要があります。
6. NAT=Network Address Translation : ネットワーク アドレス変換
7. 512、768、1024、1536、または 2048 以外の鍵 (係数) サイズを指定すると、エラーが表示され、トラストポイントの設定は適用されません。鍵を生成 (同じ **key\_label** を使用) し、サポート対象の係数サイズを指定して鍵を置き換えてから、**ステップ 12** を再実行します。

次に、SSL プロキシ サービスを設定する例を示します。

```
ssl-proxy(config)# ssl-proxy service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# nat client t2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint tp1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

管理および設定する仮想 IP アドレスおよびサーバ IP アドレスが多い場合は、ワイルドカードのプロキシを設定することができます。


次に、ワイルドカードの SSL プロキシ サービスを設定する例を示します。この設定により、**proxy1** が 10.0.0.1 ~ 10.25.255.254 の仮想 IP アドレスを受け入れるようになります。

```
ssl-proxy(config)# ssl-proxy service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.0.0.0 255.0.0.0 protocol tcp port 443
secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 20.1.2.3 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

## SSLv2 転送

SSL ドータカードは、SSL version 2.0 (SSLv2) 接続を終了できません。ただし、**server** コマンドで **sslv2** キーワードを入力して、SSLv2 接続を別のサーバに転送するように SSL ドータカードを設定できます。SSLv2 サーバの IP アドレスを設定すると、SSL ドータカードはすべての SSLv2 接続をそのサーバに透過的に転送します。SSLv2 転送が必要な場合は、SSL version 3.0 (SSLv3) または Transport Layer Security (TLS) バージョン 1.0 接続のオフロードに使用されるサーバの IP アドレスに加えて、SSLv2 サーバの IP アドレスを設定する必要があります。

SSLv2 転送を設定する手順は、次のとおりです。

コマンド	目的
<pre>ssl-proxy(config-ssl-proxy)# <b>server ipaddr ip_addr protocol tcp port port sslv2<sup>1</sup></b></pre>	<p>プロキシのターゲット サーバの IP アドレス、ポート番号、およびトランスポート プロトコルを定義します。</p> <p> (注) ターゲット サーバの IP アドレスには、SLB デバイスの仮想 IP アドレス、または Web サーバの実 IP アドレスを指定できます。</p>

1. **sslv2** キーワードを入力して、SSLv2 クライアント接続を SSLv2 サーバに転送します。**sslv2** を入力する場合、別のサーバ IP アドレスを設定して、SSLv3 または TLS バージョン 1.0 接続をオフロードします。

次に、SSL プロキシ サービスを設定して、SSLv2 接続を転送する例を示します。

```
ssl-proxy(config)# ssl-proxy service frontend
ssl-proxy(config-ssl-proxy)# virtual ipaddr 35.200.200.102 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.2 protocol tcp port 443 sslv2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

## SSL クライアント プロキシ サービス

SSL クライアント プロキシ サービスの設定では、プロキシ サービスがクリア テキスト トラフィックを受け入れ、このトラフィックを SSL トラフィックに暗号化し、バックエンドの SSL サーバに転送するように指定します。

SSL サーバ プロキシの証明書を設定する必要がありますが、SSL クライアント プロキシの証明書を設定する必要がありません。SSL クライアント プロキシの証明書を設定すると、サーバがハンドシェイク プロトコルのクライアント認証フェーズの間に送信する証明書要求メッセージに応じて、この証明書が送信されます。



(注)

SSL ポリシーは、SSL クライアント プロキシ サービスでは **server** サブコマンドで設定され、SSL サーバ プロキシ サービスでは **virtual** サブコマンドで設定されます。

SSL クライアント プロキシ サービスを設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy(config)# ssl-proxy service proxy_name client</code>	SSL プロキシ サービスの名前を定義します。 <b>client</b> キーワードは、SSL クライアント プロキシ サービスを設定します。   (注) <code>proxy-name</code> の値は、大文字と小文字が区別されます。
ステップ 2	<code>ssl-proxy(config-ssl-proxy)# virtual ipaddr ip_addr [mask_addr]<sup>1</sup> protocol tcp port port secondary</code>	CSM-S がプロキシとして機能する仮想サーバの IP アドレス、トランスポート プロトコル (TCP) およびポート番号を定義します。   (注) <b>secondary</b> キーワードは、必須です。
ステップ 3	<code>ssl-proxy(config-ssl-proxy)# server ipaddr ip_addr protocol tcp port port</code>	プロキシのターゲット サーバの IP アドレス、ポート番号、およびトランスポート プロトコルを定義します。   (注) ターゲット サーバの IP アドレスには、SLB デバイスの仮想 IP アドレス、または Web サーバの実 IP アドレスを指定できます。
ステップ 4	<code>ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp_policy_name<sup>2</sup></code>	(任意) プロキシ サーバのクライアント側に TCP ポリシーを適用します。TCP ポリシーのパラメータについては、「TCP ポリシーの設定」(p.7-13) を参照してください。
ステップ 5	<code>ssl-proxy(config-ssl-proxy)# server policy ssl ssl_policy_name<sup>2</sup></code>	(任意) プロキシ サーバのサーバ側に SSL ポリシーを適用します。SSL ポリシーのパラメータについては、「SSL ポリシーの設定」(p.7-11) を参照してください。
ステップ 6	<code>ssl-proxy(config-ssl-proxy)# server policy tcp tcp_policy_name</code>	(任意) プロキシ サーバのサーバ側に TCP ポリシーを適用します。「TCP ポリシーの設定」(p.7-13) を参照してください。

	コマンド	目的
ステップ 7	<code>ssl-proxy(config-ssl-proxy)# policy http-header http_header_policy_name</code>	(任意) プロキシ サーバに HTTP ヘッダー ポリシーを適用します。「 <a href="#">HTTP ヘッダー挿入</a> 」(p.7-14) を参照してください。
ステップ 8	<code>ssl-proxy(config-ssl-proxy)# policy url-rewrite url_rewrite_policy_name</code>	(任意) URL リライト ポリシーを適用します。「 <a href="#">URL リライトの設定</a> 」(p.7-17) を参照してください。
ステップ 9	<code>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</code>	信頼できる認証局プールとプロキシ サービスを関連付けます。認証局プールについては、「 <a href="#">クライアントの証明書の認証</a> 」(p.8-47) を参照してください。
ステップ 10	<code>ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only<sup>3</sup>   all<sup>4</sup>}</code>	クライアントの証明書認証をイネーブルにして、検証の形式を指定します。クライアントの証明書認証については、「 <a href="#">クライアントの証明書の認証</a> 」(p.8-47) を参照してください。
ステップ 11	<code>ssl-proxy(config-ssl-proxy)# nat {server   client natpool_name}</code>	(任意)SSL ドータカードによってオープンされたサーバ側の接続に、サーバ NAT <sup>5</sup> またはクライアント NAT を使用するよう指定します。「 <a href="#">NAT の設定</a> 」(p.7-25) を参照してください。
ステップ 12	<code>ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint trustpoint_label</code>	(任意) トラストポイントの設定をクライアント プロキシに適用します。  (注) トラストポイントは認証局サーバ、鍵パラメータと鍵生成方法、およびプロキシサーバの証明書登録方法を定義します。トラストポイントの設定手順については、「 <a href="#">トラストポイントの宣言</a> 」(p.8-7) を参照してください。
ステップ 13	<code>ssl-proxy(config-ssl-proxy)# inservice</code>	プロキシサーバを管理上のアップに設定します。

1. マスク アドレスを設定して、ワイルドカードのプロキシ サービスを指定します。ワイルドカードのプロキシ サービスを設定するには、**secondary** キーワードを入力する必要があります。
2. パラメータを指定しないでポリシーを作成すると、デフォルト値を使用してポリシーが作成されます。
3. **signature-only** を確認する場合、信頼される認証局プールの信頼される認証局トラストポイントの 1 つに対応するレベルで、認証が停止します。
4. すべてを検証する場合は、証明書チェーンの最上位の発行元が信頼できる認証局トラストポイントとして設定されていなければなりません。SSL ドータカードは、ピア証明書チェーンに含まれるすべての証明書を認証し、最上位の認証局でのみ停止します。最上位の認証局に対応する認証局トラストポイントがなければなりません。また、このトラストポイントを認証する必要があります。
5. NAT=Network Address Translation : ネットワーク アドレス変換

次に、SSL クライアント プロキシ サービスを設定する例を示します。

```
ssl-proxy(config)# ssl-proxy service proxy1 client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy ssl ssl1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```



## NAT の設定

クライアント接続では、クライアントが接続要求を送信し、SSL ドータカードがこれを受信します。サーバ接続では、SSL ドータカードが接続要求を送信します。

サーバ接続にはクライアント NAT、サーバ NAT、またはその両方を設定できます。



(注) SSL ドータカードにクライアント NAT が設定されている場合、CSM 側にも設定し、機能させる必要があります。

## サーバ NAT

`ssl-proxy service` コマンドを使用して設定されたサーバ IP アドレスは、SSL ドータカードがプロキシとして機能する、SSL ドータカードまたは実サーバのいずれかの宛先装置の IP アドレスとポートを指定します。サーバ NAT を設定した場合、サーバ IP アドレスはサーバ接続の宛先 IP アドレスとして使用されます。サーバ NAT が設定されていない場合、サーバ接続の宛先 IP アドレスは SSL ドータカードがプロキシ対象とする **virtual ipaddress** と同じです。SSL ドータカードは常に、**server ipaddress** サブコマンドに入力されたポート番号を使用して、ポート変換を行います。

サーバ NAT を設定する手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy (config)# <b>ssl-proxy service</b> ssl_proxy_name</code>	SSL プロキシ サービスを定義します。
ステップ 2	<code>ssl-proxy (config-ssl-proxy)# <b>nat server</b></code>	指定されたサービスの SSL オフロード サーバ接続に対して、NAT サーバアドレスをイネーブルにします。

## クライアント NAT

クライアント NAT を設定する場合、サーバ接続の送信元 IP アドレスおよびポートは NAT プールから取得されます。クライアント NAT が設定されていない場合、サーバ接続の送信元 IP アドレスおよびポートは、クライアント接続の送信元 IP アドレスおよび送信元ポートから取得されます。

SSL ドータカードがサポートする合計接続数 (256,000 接続) が満たされるように、十分な数の IP アドレスを割り当ててください。各 IP アドレスに 32,000 個のポートを割り当てる場合は、NAT プールに 8 個の IP アドレスを設定します。SSL ドータカードがサポートする合計接続数に必要な IP アドレスよりも少ない IP アドレスを設定しようとすると、拒否されます。

NAT プールを設定して、プロキシ サービスに NAT プールを割り当てる手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy (config)# <b>ssl-proxy natpool</b> natpool_name start_ip_addr end_ip_addr netmask</code>	SSL ドータカードがクライアント NAT を実装するために使用する IP アドレスのプールを定義します。
ステップ 2	<code>ssl-proxy (config-ssl-proxy)# <b>ssl-proxy service</b> ssl_proxy_name</code>	SSL プロキシ サービスを定義します。
ステップ 3	<code>ssl-proxy (config-ssl-proxy)# <b>nat client</b> natpool_name</code>	指定されたサービスの SSL オフロード サーバ接続に使用されるクライアントアドレス用の NAT プールを設定します。

## TACACS、TACACS+、および RADIUS の設定

TACACS、TACACS+、および RADIUS の設定手順については、次の URL を参照してください。

- 『Cisco IOS Security Configuration Guide, Release 12.2』の「Configuring RADIUS」の章  
[http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgr/fsecur\\_c/fsecp/scfrad.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgr/fsecur_c/fsecp/scfrad.htm)
- 『Cisco IOS Security Configuration Guide, Release 12.2』の「Configuring TACACS+」の章  
[http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgr/fsecur\\_c/fsecp/scftplus.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgr/fsecur_c/fsecp/scftplus.htm)

## SNMP トラップの設定

サポート対象の MIB (管理情報ベース) のリストについては、次の URL を参照してください。



<http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml>



(注)

CSM-S のシスコ製品の MIB ID は、ciscoproducs.610 です。ciscoproducs.554 の SSLSM とは異なりますので注意してください。

SNMP (簡易ネットワーク管理プロトコル) トラップをイネーブルにする手順は、次のとおりです。

	コマンド	目的
ステップ 1	<code>ssl-proxy(config)# snmp-server host addr traps version version ssl-proxy</code>	トラップの送信先となる、外部ネットワーク管理装置の IP アドレスを指定します。
ステップ 2	<code>ssl-proxy(config)# snmp-server enable traps ssl-proxy cert-expiring</code>	(任意) SSL プロキシ証明書の期限切れ通知トラップをイネーブルにします。   (注) 証明書の期限確認の間隔を <b>0</b> に設定すると、期限切れの通知トラップが送信されません。証明書の期限切れ警告のイネーブル化については、「 <a href="#">証明書の有効期限に関する警告の設定</a> 」(p.8-44) を参照してください。   (注) 期限切れの通知トラップは、現在設定されているプロキシサービスの証明書だけに送信されます。
ステップ 3	<code>ssl-proxy(config)# snmp-server enable traps ssl-proxy oper-status</code>	(任意) SSL プロキシの動作ステータス通知トラップをイネーブルにします。
ステップ 4	<code>ssl-proxy(config)# snmp-server queue-length length</code>	(任意) キューが空になる前に、保持されるトラップイベント数を指定します。デフォルトの <i>length</i> は 10 です。有効値は、1 ~ 1000 です。
ステップ 5	<code>ssl-proxy# show snmp</code>	SNMP 情報を表示します。

次に、SNMP トラップをイネーブルにする例を示します。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# snmp-server host 10.1.1.1 traps version 2c ssl-proxy
ssl-proxy(config)# snmp-server enable traps ssl-proxy cert-expiring
*Nov 27 03:47:10.739:%STE-6-PROXY_CERT_EXPIRING_TRAP_ENABLED:SNMP trap for proxy
service
certificate expiration warning has been enabled.
ssl-proxy(config)# snmp-server enable traps ssl-proxy oper-status
*Nov 27 03:46:59.607:%STE-6-PROXY_OPER_STATUS_TRAP_ENABLED:SNMP trap for proxy service
operational status change has been enabled.
ssl-proxy(config)# snmp-server queue-length 256
ssl-proxy(config)# end

ssl-proxy# show snmp
0 SNMP packets input
  0 Bad SNMP version errors
  0 Unknown community name
  0 Illegal operation for community name supplied
  0 Encoding errors
  0 Number of requested variables
  0 Number of altered variables
  0 Get-request PDUs
  0 Get-next PDUs
  0 Set-request PDUs
8 SNMP packets output
  0 Too big errors (Maximum packet size 1500)
  0 No such name errors
  0 Bad values errors
  0 General errors
  0 Response PDUs
  8 Trap PDUs

SNMP logging:enabled
  Logging to 10.1.1.1.162, 0/256, 0 sent, 0 dropped.
ssl-proxy#
```

## 暗号セルフテストのイネーブル化



(注) 電源投入時暗号チップセルフテストおよび鍵テストは、起動時に1回のみ行われます。



(注) セルフテストはトラブルシューティング専用です。このテストを実行すると、実行時のパフォーマンスに影響します。

セルフテストを実行する手順は、次のとおりです。

コマンド	目的
<code>ssl-proxy(config)# ssl-proxy crypto self-test time-interval time</code>	暗号セルフテストをイネーブルにします。 <i>time</i> のデフォルト値は3秒です。有効範囲は1～8です。

次に、暗号セルフテストをイネーブルにして、暗号情報を表示する例を示します。

```
ssl-proxy(config)# ssl-proxy crypto self-test time-interval 1
ssl-proxy(config)# end
```

## 統計情報の表示

統計情報を表示する手順は、次のとおりです。

コマンド	目的
<code>ssl-proxy(config)# show ssl-proxy stats {crypto   hdr   ipc   pki [auth   cache   cert-header   database   expiring   history   ipc   memory]   service   ssl   tcp   url}</code>	指定された統計情報を表示します。

次に、ヘッダー挿入情報を表示する例を示します。

```
ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
  Session Headers Inserted :1          Custom Headers Inserted :0
  Session Id's Inserted    :2          Client Cert. Inserted   :0
  Client IP/Port Inserted  :0
  No End of Hdr Detected   :0          Payload no HTTP header  :0
  Desc Alloc Failed        :0          Buffer Alloc Failed      :0
  Client Cert Errors       :0          No Service               :0
```

次に、暗号情報を表示する例を示します。

```
ssl-proxy# show ssl-proxy stats crypto
Crypto Statistics from SSL Module:1
Self-test is running
Current device index is 1
Time interval between tests is 1 seconds
Device 0 statistics:
Total Number of runs:50
Runs all passed:50
Number of timer error:0
-----
Test Name                Passed  Failed  Did-not-run
-----
 0 Power-on Crypto chip sel    1      0      0
 1 Power-on Crypto chip key    1      0      0
 2 Hash Test Case 1           50     0      0
 3 Hash Test Case 2           50     0      0
 4 Hash Test Case 3           50     0      0
 5 Hash Test Case 4           50     0      0
 6 SSL3 MAC Test Case 1       50     0      0
 7 SSL3 MAC Test Case 2       50     0      0
 8 TLS1 MAC Test Case 1       50     0      0
 9 TLS1 MAC Test Case 2       50     0      0
10 DES Server Test            50     0      0
11 DES Encrypt Test 1         50     0      0
12 DES Decrypt Test 1         50     0      0
13 DES Encrypt Test 2         50     0      0
14 DES Decrypt Test 2         50     0      0
15 ARC4 Test Case 1           50     0      0
16 ARC4 Test Case 2           50     0      0
17 ARC4 Test Case 3           50     0      0
18 ARC4 State Test Case 1     50     0      0
19 ARC4 State Test Case 2     50     0      0
20 ARC4 State Test Case 3     50     0      0
21 ARC4 State Test Case 4     50     0      0
22 HMAC Test Case 1           50     0      0
23 HMAC Test Case 2           50     0      0
24 Random Bytes Generation    50     0      0
25 RSA Encrypt/Decrypt Test   50     0      0
26 Master Secret Generation   50     0      0
27 Key Material Generation    50     0      0
28 SSL3 Handshake Hash Test   50     0      0
29 TLS1 Handshake Hash Test   50     0      0

Device 1 statistics:
Total Number of runs:49
Runs all passed:49
Number of timer error:0
-----
Test Name                Passed  Failed  Did-not-run
-----
 0 Power-on Crypto chip sel    1      0      0
 1 Power-on Crypto chip key    1      0      0
 2 Hash Test Case 1           50     0      0
 3 Hash Test Case 2           50     0      0
 4 Hash Test Case 3           50     0      0
 5 Hash Test Case 4           50     0      0
 6 SSL3 MAC Test Case 1       50     0      0
 7 SSL3 MAC Test Case 2       50     0      0
 8 TLS1 MAC Test Case 1       50     0      0
 9 TLS1 MAC Test Case 2       50     0      0
10 DES Server Test            50     0      0
11 DES Encrypt Test 1         50     0      0
12 DES Decrypt Test 1         50     0      0
13 DES Encrypt Test 2         50     0      0
14 DES Decrypt Test 2         50     0      0
15 ARC4 Test Case 1           50     0      0
16 ARC4 Test Case 2           50     0      0
```

17	ARC4 Test Case 3	50	0	0
18	ARC4 State Test Case 1	49	0	0
19	ARC4 State Test Case 2	49	0	0
20	ARC4 State Test Case 3	49	0	0
21	ARC4 State Test Case 4	49	0	0
22	HMAC Test Case 1	49	0	0
23	HMAC Test Case 2	49	0	0
24	Random Bytes Generation	49	0	0
25	RSA Encrypt/Decrypt Test	49	0	0
26	Master Secret Generation	49	0	0
27	Key Material Generation	49	0	0
28	SSL3 Handshake Hash Test	49	0	0
29	TLS1 Handshake Hash Test	49	0	0

次に、PKI 証明書認証および許可の統計情報を表示する例を示します。

```
ssl-proxy# show ssl-proxy stats pki auth
Authentication request timeout:240 seconds
Max in process:100 (requests)
Max queued before dropping:0 (requests)
Certificate Authentication & Authorization Statistics:
  Requests started:2
  Requests finished:2
  Requests pending to be processed:0
  Requests waiting for CRL:0
  Signature only requests:0
  Valid signature:0
  Invalid signature:0
  Total number of invalid certificates:0
  Approved with warning (no crl check):2
  Number of times polling CRL:0
  No certificates present:0
  Failed to get CRL:0
  Not authorized (e.g. denied by ACL):0
  Root certificates not self-signed:0
  Verify requests failed (e.g. expired or CRL operation failed):0
  Unknown failure:0
  Empty certificate chain:0
  No memory to process requests:0
  DER encoded certificates missing:0
  Bad DER certificate length:0
  Failed to get key from certificate:0
  Issuer CA not in trusted CA pool:0
  Issuer CA certificates not valid yet:0
  Expired issuer CA certificates:0
  Peer certificates not valid yet:0
  Expired peer certificates:0
```

次に、PKI ピア証明書キャッシュの統計情報を表示する例を示します。

```
ssl-proxy# show ssl-proxy stats pki cache
Peer certificate cache size:0 (entries), aging timeout:30 (minutes)
Peer certificate cache statistics:
  In use:0 (entries)
  Cache hit:0
  Cache miss:0
  Cache allocated:0
  Cache freed:0
  Cache entries expired:0
  Cache error:0
  Cache full (wrapped around):0
  No memory for caching:0
```

次に、転送データ ユニットの統計情報を表示する例を示します。

```
ssl-proxy# show ssl-prox stats fdu
FDU Statistics:
  IP Frag Drops           : 0           IP Version Drops      : 0
  IP Addr Discards        : 0           Serv_Id Drops         : 0
  Conn Id Drops           : 0           Bound Conn Drops     : 0
  Vlan Id Drops           : 0           TCP Checksum Drops   : 0
  Hash Full Drops         : 0           Hash Alloc Fails     : 0
  Flow Creates            : 536701       Flow Deletes          : 536701
  Conn Id allocs          : 268354       Conn Id deallocs     : 268354
  Tagged Pkts Drops       : 0           Non-Tagg Pkts Drops  : 0
  Add ipcs                : 3           Delete ipcs           : 0
  Disable ipcs            : 1           Enable ipcs           : 0
  Unsolicited ipcs        : 1345        Duplicate Add ipcs   : 0
  IOS Broadcast Pkts      : 43432       IOS Unicast Pkts     : 12899
  IOS Multicast Pkts      : 0           IOS Total Pkts       : 56331
  IOS Congest Drops       : 0           SYN Discards         : 0
FDU Debug Counters:
  Inv. Conn Drops         : 0           Inv. Conn Pkt Drops  : 0
  Inv. TCP opcodes        : 0
  Inv. Fmt Pkt Drops      : 0           Inv. Bad Vlan ID     : 0
  Inv. Bad Ctl Command    : 0           Inv. TCP Congest     : 0
  Inv. Bad Buffer Fmt     : 0           Inv. Buf Undersized  : 0
ssl-proxy#
```

## クラッシュ情報の収集

クラッシュ情報機能は、ソフトウェア強制リセットを修正する開発者に必要な情報を収集します。ソフトウェア強制リセット情報を収集するには、**show ssl-proxy crash-info** コマンドを入力します。ソフトウェア強制リセットを複数回行った場合は、最後のクラッシュ情報のみが表示されます。**show ssl-proxy crash-info** コマンドが情報収集プロセスを完了するまでに、1～6分かかります。



(注) **show stack** コマンドは、SSL ドータカードのソフトウェア強制リセット情報を収集するためにサポートされているコマンドではありません。

次に、ソフトウェア強制リセット情報を収集する例を示します。

```
ssl-proxy# show ssl-proxy crash-info

===== SSL daughter card - START OF CRASHINFO COLLECTION =====

----- COMPLEX 0 [FDU_IOS] -----

NVRAM CHKSUM:0xEB28
NVRAM MAGIC:0xC8A514F0
NVRAM VERSION:1

+++++++ CORE 0 (FDU) ++++++

  CID:0
  APPLICATION VERSION:2003.04.15 14:50:20 built for cantuc
  APPROXIMATE TIME WHEN CRASH HAPPENED:14:06:04 UTC Apr 16 2003
  THIS CORE DIDN'T CRASH
  TRACEBACK:222D48 216894
  CPU CONTEXT -----

$0 :00000000, AT :00240008, v0 :5A27E637, v1 :000F2BB1
a0 :00000001, a1 :0000003C, a2 :002331B0, a3 :00000000
t0 :00247834, t1 :02BF8BA0, t2 :02BF8BB0, t3 :02BF8BA0
t4 :02BF8BB0, t5 :00247834, t6 :00000000, t7 :00000001
s0 :00000000, s1 :0024783C, s2 :00000000, s3 :00000000
s4 :00000001, s5 :0000003C, s6 :00000019, s7 :0000000F
t8 :00000001, t9 :00000001, k0 :00400001, k1 :00000000
gp :0023AE80, sp :031FFF58, s8 :00000019, ra :00216894
LO :00000000, HI :0000000A, BADVADDR :828D641C
EPC :00222D48, ErrorEPC :BFC02308, SREG :34007E03
Cause 0000C000 (Code 0x0):Interrupt exception

CACHE ERROR registers -----

CacheErrI:00000000, CacheErrD:00000000
ErrCtl:00000000, CacheErrDPA:0000000000000000

PROCESS STACK -----
  stack top:0x3200000

Process stack in use:

sp is close to stack top;

printing 1024 bytes from stack top:

031FFC00:06405DE0 002706E0 0000002D 00000001  .@]`.'.`.....
031FFC10:06405DE0 002706E0 00000001 0020B800  .@]`.'.`..... 8.
031FFC20:031FFC30 8FBF005C 14620010 24020004  ..|0.?.\..b..$...
.....
.....
.....
FFFFFFD0:00000000 00000000 00000000 00000000  .....
FFFFFFE0:00627E34 00000000 00000000 00000000  .b~4.....
FFFFFFF0:00000000 00000000 00000000 00000006  .....

===== SSL daughter card - END OF CRASHINFO COLLECTION =====
```



## VTS デバッグのイネーブル化

モジュール上の異なるプロセッサ (FDU、TCP、SSL) をデバッグするために、SSL ドータカードに Virtual Terminal Server (VTS; 仮想端末サーバ) が搭載されています。



(注) TCP デバッグ コマンドは、負荷がほとんどない場合 (仮想サーバまたは実サーバに接続が確立されていない場合など) に、基本的な接続問題のトラブルシューティングを行うときのみ使用してください。

TCP デバッグ コマンドを使用すると、TCP モジュールはコンソールにデバッグ情報を大量に表示するため、これによってモジュールのパフォーマンスが大幅に低下することがあります。モジュールのパフォーマンスが低いと、TCP 接続タイマー、パケット、およびステート移行の処理に遅延が生じることがあります。

SSL ドータカードの FDU (ポート 2001)、TCP (ポート 2002)、および SSL (ポート 2003) プロセッサに到達できるように、ワークステーションまたは PC からモジュールの VLAN IP アドレスの 1 つに Telnet 接続を確立します。

デバッグ情報を表示する手順は、次のとおりです。

コマンド	目的
ssl-proxy# [no] <b>debug ssl-proxy</b> {fdu   ssl   tcp} [type]	指定されたシステム コンポーネントのデバッグ フラグをオンまたはオフにします。

Telnet 接続の確立後、SSL 証明書管理コンソールから **debug ssl-proxy {tcp | fdu | ssl}** コマンドを入力します。クライアントから接続が 1 つ送信されて、TCP コンソールにログが表示されます。

次に、接続の TCP 状態のログを表示して、デバッグ状態を確認する例を示します。

```
ssl-proxy# debug ssl-proxy tcp state
ssl-proxy# show debugging
STE Mgr:
  STE TCP states debugging is on
```

次に、ワークステーションまたは PC の出力例を示します。

```
Conn 65066 state CLOSED --> state SYN_RECEIVED
Conn 65066 state SYN_RECEIVED --> state ESTABLISHED
Conn 14711 state CLOSED --> state SYN_SENT
Conn 14711 state SYN_SENT --> state ESTABLISHED
Conn 14711 state ESTABLISHED --> state CLOSE_WAIT
Conn 65066 state ESTABLISHED --> state FIN_WAIT_1
Conn 65066 state FIN_WAIT_1 --> state FIN_WAIT_2
Conn 65066 state FIN_WAIT_2 --> state TIME_WAIT
Conn 14711 state CLOSE_WAIT --> state LAST_ACK
Conn 14711 state LAST_ACK --> state CLOSED
#####Conn 65066 state TIME_WAIT --> state CLOSED
```

## ■ VTS デバッグのイネーブル化